4.1.0.0

Milestone 4.1

Enhancements

- **Smoother rotational motion blur**: Interpolation between two motion blur keys now better accounts for rotation. This improvement is very obvious when rotating a propeller blade by close to 180 degrees, since now the blade properly follows the curved path. We can only interpolate angles of up to 180 degrees, so to get a 360 degree rotation (or more), will still take multiple keys. However, the number of keys required can now be greatly reduced. This also prevents certain crashes from occurring where the less precise interpolation caused degenerate non-invertible matrices to be produced. This feature can be turned off by setting the options variable curved_motionblur to false. If we do not hear of any problems with this new feature (which is enabled by default), we will eventually get rid of the curved_motionblur option and make this the only option. So please let us know if you encounter issues with this. (#3604)
- **Rolling shutter**: It is now possible to simulate the type of rolling shutter effect seen in footage shot with digital cameras that use CMOS-based sensors such as Blackmagics, Alexas, REDs, and even iPhones. This is controlled by the new rolling_shutter camera enum parameter, which defaults to "off" and can be set to "top" (top-to-bottom being the most common scanning direction), "bottom", "left" or "right". (#3127)
- **Texture support in mesh_light**: In a similar way to other mappable lights, a mesh light's color attribute can now be linked to a texture map. A single color will be assigned per mesh triangle, so the resolution of the mesh can have an effect on the quality of the illumination. (#3552)
- **Reduced per-thread memory footprint** : Reduced the memory pool overhead. Savings are most noticeable for large AA sampling rates and large bucket sizes. For an 8x8 AA samples, and 64x64 bucket size render, about 5.3MB is saved per thread. For 32 threads, that is 170MB saved. Also improved the accuracy for memory reporting of memory pools. (#3594)
- **Faster multi-layered SSS**: It is now possible to compute a weighted sum of an arbitrary number of SSS profiles in a single function call via importance sampling, without having to fire a new set of samples for each diffusion radius. This allows for a result that is similar in quality to multiple invocations of the classic cubic/gaussian SSS lookups, but at a fraction of the cost. The speedup is linear on the number of layers. (#3640)
- **Faster .ass file load**: Both .ass and .ass.gz files should now load 2x to 3x faster. (#3532, #3570, #3579, #3581)
- **Reduced per-pixel overhead**: We have added a number of optimizations to the processing, storage and filtering of subpixel samples, AOVs, and camera ray creation. The result is a slight speedup for all renders, and a more significant speedup for pixels that have simpler geometry and/or shaders. For example, a mostly empty black frame with no AOVs now renders 2x faster, and the same scene but using 20 AOVs renders 6x faster. (#3585, #3602, #3614, #3618, #3633)
- **Faster light sampling**: The sampling of spot lights and area lights has been optimized. This can result in a 15% speedup in scenes with many lights. (#3617, #3683)
- **Faster physical_sky**: The physical_sky shader has been optimized and is now 30% faster. In practice, this speedup only applies to the precomputation of the importance tables of the skydome_light, specially when large importance table resolutions are used (4k and higher) as is often necessary to capture the small sun disk of the physical sky. (#3529)
- **Faster skydome_light**: A performance regression was introduced in Arnold 4.0.16, where texture mapped skydome lights would unnecessarily trace shadow rays to black parts of the sky. This is now fixed, and performance should be back to pre-4.0.16 levels. (#3555)
- **Faster back diffuse**: The Kb (backlighting) section of the standard shader is now more performant in some cases. When used in conjunction with both front diffuse and specular it will cast one-third fewer shadow rays. It also got a minor speed bump by not evaluating shader networks attached to Kb more than once. (#3569)
- **Faster adaptive subdivision**: Several small optimizations in the subdivision engine have resulted in an aggregated 10% speedup when using adaptive subdivision. (#3547)
- **Cached per-component shader links**: Redundant shader evaluations that occurred when the same shader is attached to multiple components of another shader's input (e.g. to Kd_color.r, g, and b) have now been eliminated. This can lead to significant speedups in complex shader networks that abused per-component linking, something that has been observed in the workflow of some Maya artists. (#3657)
- **Volume ray differentials**: Gobo textures applied to spotlights now make use of mipmapping and mipmap_bias, therefore reducing the amount of texture I/O. (#3067)
- **Raised texture cache to 1GB and texture_max_sharpen to 1.5**: The default value for options.texture_max_memory_MB was 512 MB which is now likely too small for complex production shots and could cause cache thrashing. We have raised this default to 1 GB. Very complex film projects may want to increase this manually to 2 or even 4 GB. We have also raised the default max_texture_sharpen from 1.0 to 1.5. This will produce sharper, nicer looking textures, but it will also result in increased texture usage -- sometimes by around 2x. If the increased texture usage causes performance problems, this can be manually lowered back to 1.0. (#3584)
- **objwire utility shading mode**: This new mode in the utility shader combines polywire and obj shading in one mode. (#3676)
- **uniformid utility shading mode**: This new mode in the utility shader allows to color by patch instead of by polygon, by curve instead of curve segment. This mirrors how uniform user data is looked up. (#3686)
- **Smart multi-platform shared libraries file extensions**: Shared libraries loaded through AiLoadPlugins() or through the dso parameter in proceduralnodes now accept all regular platform file extensions (.dll, .so, .dylib) regardless of the host platform. Custom extensions are also allowed. (#3432)
- **Report C/C++ standard libraries in Linux**: The version/build information in the log file was incorrectly reporting the version of the GNU C standard library (glibc) installed in the system instead of the version with which Arnold was built. In addition, we now report the version of the GNU C++ library (libstdc++) with which Arnold was built. (#3553)
- **Remove useless matrix warnings**: To avoid confusion and polluting log files, we no longer issue performance warnings when identity matrices are supplied or when the motion blur matrices are identical. The overhead when this occurs is insignificant. Note that these matrices are silently removed to ensure optimal performance during ray tracing. (#3650)
- **Upgraded OIIO to 1.2.2**: This OpenImageIO upgrade comes with several bug fixes, more accurate texture filtering, faster texture I/O and up to 10x fastermaketx processing for large images. It also no longer exports the OIIO symbols, see Incompatible changes below for more details. (#3514, #3645, #2898, #3655)
- **Upgraded RLM to 10.1BL2**: We have upgraded the license server and the external library controlling the licensing subsystem from version 9.4BL4 to 10.1BL2, a more stable release fixing various crashes, bugs, hangs and memory leaks. (#3619)

API additions

- **Return values for AiArraySet***: The AiArraySet* API functions now return false when they fail. This lets applications/exporters print out more useful information about the context in which the failure occurred. (#2684)
- **Return value for AiUniverseCacheFlush**: This function now returns an error code, in particular it will return false when called during a render in progress. (#3361)
- **binary argument in AiASSWrite()**: A new binary argument has been added to AiASSWrite() which controls whether arrays use binary encoding when writing an .ass file. The old global option binary_ass which used to serve this purpose has been removed. Since -set options. binary_ass off no longer works, we have added a new kick option to disable binary encoding when re-saving .ass files: kick in.ass -db -resave out.ass. (#2828)
- **Texture blur, mipmap mode, and fill**: The AtTextureParams struct now has additional parameters blur_s, blur_t, mipmap_mode, and fill. While the width parameters that were there already will multiply the filter width, the new blur parameters will add to it. The mipmap_mode is for fine-tuning MIP filtering; in some cases you know you won't need high-quality MIP filtering, and so you can choose a faster, more approximate mode instead. The fill value is used for non-existent channels in a texture, such as the alpha channel in an RGB-only texture. (#3191)
- **Light cache resets**: Shader light loops will generally automatically cache lighting data, or invalidate the cached data for a subsequent light loop if certain common items in the shader globals change (such as trace sets, surface normal, sg->fhemi, etc). Occasionally shader writers wish to get fresh lighting data for a light loop where the cache does not automatically reset; for this purpose we have provided a new API call to ensure that lighting data is recomputed. Note that calling this unnecessarily will obviously defeat the cache and impact performance, so it should only be used as strictly necessary. (#3557)

```
AI_API void AiLightsResetCache(AtShaderGlobals* sg);
```

- **Surface normal derivatives**: Added the surface normal derivatives with respect to image space (dNdx and dNdy) to the AtShaderGlobals struct. (#3193)
- **AiShaderGlobalsGetPolygon() with 256+ vertices**: The existing AiShaderGlobalsGetPolygon() API function now works on polygons with more than 255 vertices. If a NULL pointer is passed as vertex array, the function will simply return the number of vertices in the polygon. This way, an array could be allocated for the right number of vertices (using AiShaderGlobalsQuickAlloc() for speed) and passed in a second call toAiShaderGlobalsGetPolygon() to store the actual vertices into the array. (#3468)
- **AiColor()**: To save a few keystrokes, a new API function AiColor() has been added which is exactly the same as the older AiColorCreate() but shorter. AiColorCreate() is now deprecated and will be removed in the next API-breaking release. (#3653)
- **DriverNeedsBucket() and DriverProcessBucket()**: All custom output drivers must implement these two additional methods. DriverNeedsBucket()tells the renderer whether a driver needs a bucket to be rendered (perhaps because the bucket is already available) and DriverProcessBucket() gives the custom driver a chance to preprocess a bucket in a non-locking way. (#3627)
- **AiNodeSetAttributes() in Python**: When this function was introduced in the C API, we forgot to add the corresponding Python binding. (#3586)

Incompatible changes

- **Changes in .ass files**: The size of binary-encoded .ass files has been reduced slightly. Among other changes, binary-encoded arrays are now written as a single big line of ASCII characters. This new Arnold version will read older .ass files which used the previous multiline format, but please note that the newly written .ass files will not be readable by older Arnold versions, and in fact may crash while parsing. (#3542, #3609)
- **Reversed latlong and angular environment mapping**: The u axis of AiMappingLatLong(), AiMappingLatLongDerivs(), AiMappingAngularMap() and AiMappingAngularMapDerivs() have been flipped so that these projections behave as expected of environment maps. (#3577)
- **Setting light samples to 0 disables the light**: This is more consistent with how every other sampling knob works. Previously the samples were silently clamped to 1. (#3687)
- **Corrected AiNodeIsLinked() and AiNodeGetLink() behavior**: The behaviour of these functions has been modified to remove ambiguities related to per-component links. When an RGB parameter was linked with AiNodeLink(image, "Kd_color", lambert), querying the link stored in the individual R, G and B components of the parameter would incorrectly return the node linked in the parameter itself. With the new behaviour, a call toAiNodeIsLinked(lambert, "Kd_color.r") will return false, and a call to AiNodeGetLink(lambert, "Kd_color.r") will return NULL. (#2699)
- **Removed deprecated AiUniverse functions**: The legacy API functions AiUniverseGetLights, AiUniverseGetNumLights, AiUniverseGetNumGObjects, AiGetNumInstalledNodes and AiNodeEntryLookUpByIndex have been removed. Use the AtNodeIterator andAtNodeEntryIterator API instead. (#1940)
- **Removed external library symbols**: On linux and osx, OpenImageIO, TBB, and all other external libraries no longer have their symbols exported by Arnold. This means that on these platforms there can no longer be any symbol clashes with these libraries, but it also means that code that relied on these symbols being exported by Arnold must now supply their own version of these libraries. Windows still relies on a dynamic library for OIIO, but we have already removed its headers and libs and it is our intention to remove the DLL in a future version as well. (#2898)
- **Removed legacy/unused options**: We have removed a bunch of legacy global options that are no longer useful. These are AA_sampling_dither, GI_hemi_pattern, GI_Cranley_Patterson, enable_legacy_vector_noise, physically_based, auto_transparency_probabilistic, strict_procedural_compatibility, enable_memory_reporting and ignore_mis. (#3597)
- **Removed stretched-Phong BRDF**: The stretched_phong value of the specular_brdf enum in the standard shader, that has been deprecated for a couple of years, has finally been removed. (#3630)
- **Removed AiUDataSet*() functions**: These methods were not thread-safe, and were rarely used, so they caused more problems than they solved. However, no mechanism exists now to send extra data between displacement and surface shaders. (#2740)
- **Removed specialized (and slow) memory reporting support for older linux**: On older distros, such as CentOS 5, Arnold will now report 0MB memory use in the per-line memory stat of the log file. We feel this is preferable to the 10% slowdown incurred by the tons of slow system calls. (#3160)
- **Removed ambient_light**: The legacy ambient_light node has finally been removed, along with its associated API AiAmbient(). (#3656)
- **Replaced doubles with floats**: Constants, such as AI_PI, are now floats instead of doubles. Likewise, many functions that had doubles as arguments or returned doubles, now use floats. Most of these changes should be transparent, but AiSamplerGetSample() now uses float* arguments, so in order for this to compile, this function must be given floats. (#3663, #3666)

- **Removed deprecated types and functions**: The legacy typedefs of basic data types in ai_types.h (AtVoid, AtFloat, AtDouble, AtBoolean, AtInt etc) have finally been removed after being deprecated almost two years ago in Arnold 4.0. Any client code using them will therefore break, and a text replacement for the simpler built-in data types (void, float, double, bool, int etc) will be required. The deprecated legacy functions in ai_types.h are also gone, specifically BILERP(), BIHERP(), MAP01(), MAP01F(), ACOSF(), INVERSE(), AiBerpScalar(). The deprecated macros TRUE and FALSE have also been removed, which should be replaced in client code for the lowercase true and false. (#3670)
- **Removed unused AtRay fields**: The AtRay struct no longer has the lx, ly, xfrm_len, and inv_xfrm_len fields since they appear to never be used. (#3667)
- **Removed RGB AOV in favor of RGBA**: The internal, built-in AOV "RGB" has been removed in favor of always using "RGBA". Requests for the "RGB" AOV will be redirected to "RGBA" AOV, and a warning will be issued advising to use "RGBA". Also, a new parameter skip_alpha has been added to thedriver_tiff node to help preserve the old behaviour. (#3688)
- **Replaced dither_amplitude with boolean dither**: Dithering for 8-bit images can now be turned on and off in output drivers using the boolean ditherattribute. Also, the kick option -da has been removed. (#3691)
- **Removed pointcloud SSS**: Support for the legacy pointcloud-based SSS rendering method has been removed. The rendering of BSSRDFs is now only done using the much simpler ray traced method that was added in Arnold 4.0.7. This change removes the following options: sss_sample_spacing (object setting),sss_sample_distribution (object setting), sss_faceset (optional object setting), sss_threaded_sample_distribution (global setting),sss_sample_factor (global setting), show_samples (global setting), sss_subpixel_cache (global setting). (#3628)
- **Removed AiSSSPointCloudLookupCubic and AiSSSPointCloudLookupGaussian()**: These two functions have been replaced by the following more general functions, which return the weighted sum of an arbitrary number of SSS profiles instead of just three: (#3640)

```
AI_API AtColor AiBSSRDFCubic(const AtShaderGlobals* sg, const float* radius, const AtColor* weight,
unsigned int num = 1);
AI_API AtColor AiBSSRDFGaussian(const AtShaderGlobals* sg, const float* variance, const AtColor*
weight, unsigned int num = 1);
```

As an example of usage, the code snippet below would have to be changed from this:

```
AtColor sss_radius = AiShaderEvalParamRGB(p_sss_radius);
AtColor Ksss = AiShaderEvalParamRGB(p_sss_color);
AtColor sss_weight = sss_weight;
AtColor sss = AiSSSPointCloudLookupCubic(sg, sss_radius) * sss_weight;
```

To this:

```
AtColor sss_radius = AiShaderEvalParamRGB(p_sss_radius);
AtColor Ksss = AiShaderEvalParamRGB(p_sss_color);
AtColor sss_weight[3];
sss_weight[0] = AiColor(Ksss.r,      0,      0);
sss_weight[1] = AiColor(     0, Ksss.g,      0);
sss_weight[2] = AiColor(     0,      0, Ksss.b);
AtColor sss = AiBSSRDFCubic(sg,
                            &sss_radius[0], // <-- array of sample radii, starting with the first
component (red) of AtColor
                            sss_weight,     // <-- array of sample weights, one for each radius
                            3);             // <-- AiSSSLookupCubic() was actually sampling three radii
in one call
```

And here's how the sum of 6 gaussians to simulate diffusion in skin from Chapter 14 of GPU Gems 3 could be implemented:

```
   float variance[6] = {0.0064f, 0.0484f, 0.187f, 0.567f, 1.99f, 7.41f};
   AtRGB weight[6]   = {{0.233f, 0.455f, 0.649f},
                        {0.100f, 0.336f, 0.344f},
                        {0.118f, 0.198f, 0     },
                        {0.113f, 0.007f, 0.007f},
                        {0.358f, 0.004f, 0     },
                        {0.078f, 0     , 0     }};
   AtRGB sss = AiBSSRDFGaussian(sg, variance, weight, 6);
```

Bug fixes

Ticket	Summary	Component	Owner	Priority	Version	Created
#3588	crash when preprocessing polymeshes with 2 motion keys	arnold	thiago	critical	4.0	2 months
#3610	crash at render shutdown in procedurals with user-data of type array	arnold	angel	critical	4.0	5 weeks
#2699	wrong AiNodeGetLink & AiNodeIsLinked behavior	arnold	angel	major	3.3	20 months
#2740	Remove support for setting user data in displacement shaders	arnold	mike	major	4.0	20 months
#2760	64bit sizes (like memory allocations) should be of type size_t	arnold	thiago	major	3.3	19 months
#3126	Lights inside procedurals are not affected by the procedural matrix	arnold	angel	major	4.0	12 months
#3245	User data is not cloned with AiNodeClone()	arnold	angel	major	4.0	9 months
#3277	OIIO cache flushing not always works	arnold	ramon	major	4.0	8 months
#3361	AiUniverseCacheFlush() crashes while rendering	arnold	ramon	major	4.0	7 months
#3386	Transparent objects close to the ground plane of the fog atmosphere render with artifacts	arnold	alan	major	4.0	6 months
#3391	AiArraySetMtx crashes when array is NULL	arnold	marcos	major	4.0	6 months
#3425	".ass parsing (deferred)" stat was not properly accounted for	arnold	thiago	major	4.0	6 months
#3464	make AiShaderGlobalsGet*() API functions more robust towards changes in shader globals	arnold	alan	major	4.0	5 months

#3468	Fix support for polygons with >255 vertices in the API	arnold	angel	major	4.0	5 months
#3520	Random dither amplitude in output drivers is too strong	arnold	ramon	major	4.0	4 months
#3525	Arnold won't abort after exceeding max warnings/errors	arnold	thiago	major	4.0	3 months
#3534	clipping planes not affecting volumetrics	arnold	alan	major	4.0	3 months
#3537	Using AiNodeSetStr on a string array leads to a crash	arnold	angel	major	4.0	3 months
#3543	Correctly apply gamma in jpeg driver	arnold	ramon	major	4.0	3 months
#3548	BRDF sampling does not respect mesh_light orientation	arnold	ramon	major	4.0	3 months
#3549	IES parser errors cause crashes	arnold	oscar	major	4.0	3 months
#3555	skydome should take into account low_light_threshold	arnold	thiago	major	4.0	3 months
#3561	Oren-Nayar MIS functions do not work outside of a light loop	arnold	alan	major	4.0	3 months
#3562	Crash when AiNodeIsLinked() is used with an empty string	arnold	angel	major	4.0	3 months
#3577	latlong and angular environment mappings should be flipped in U	arnold	alan	major	4.0	2 months
#3586	Expose AiNodeSetAttributes() in the Python API	arnold	oscar	major	4.0	2 months
#3587	procedural_force_expand is not written out to .ass files correctly	arnold	oscar	major	4.0	2 months
#3590	IPR crash when changing bucket scanning from hilbert to something else	arnold	pal	major	4.0	2 months
#3591	bump2d crashes when used with curves	arnold	oscar	major	4.0	7 weeks
#3592	multi-element shader arrays not inherited from procedurals	arnold	angel	major	4.0	7 weeks
#3608	Crash with massive non-tiled exr driver	arnold	ramon	major	4.0	5 weeks
#3615	Clean up 'long' and 'unsigned long' usage	arnold	thiago	major	4.0	4 weeks
#3626	clip spot_light cone exactly at origin if lens_radius is > 0	arnold	ramon	major	4.0	3 weeks
#3645	OIIO reports memory wrong (windows)	oiio	ramon	major	4.0	2 weeks
#3648	Crash when a user data is declared and not defined	arnold	oscar	major	4.0	2 weeks
#3658	AtColor/AtPoint/AtPoint2 const operator[] could not be used as L-value	arnold	alan	major	4.0	11 days
#3689	EXR driver causes new threads to be spawned	arnold	ramon	major	4.0	37 hours
#3690	indirect multiplier not taken into account in skydome_light	arnold	alan	major	4.0	29 hours
#3511	crash with illegal values of curves.max_subdivs	arnold	ramon	minor	4.0	4 months
#3583	report .ass.gz being loaded when .ass doesn't exist	arnold	oscar	minor	4.0	2 months
#3638	-set does not work with matrix parameters	kick	alan	minor	4.0	2 weeks